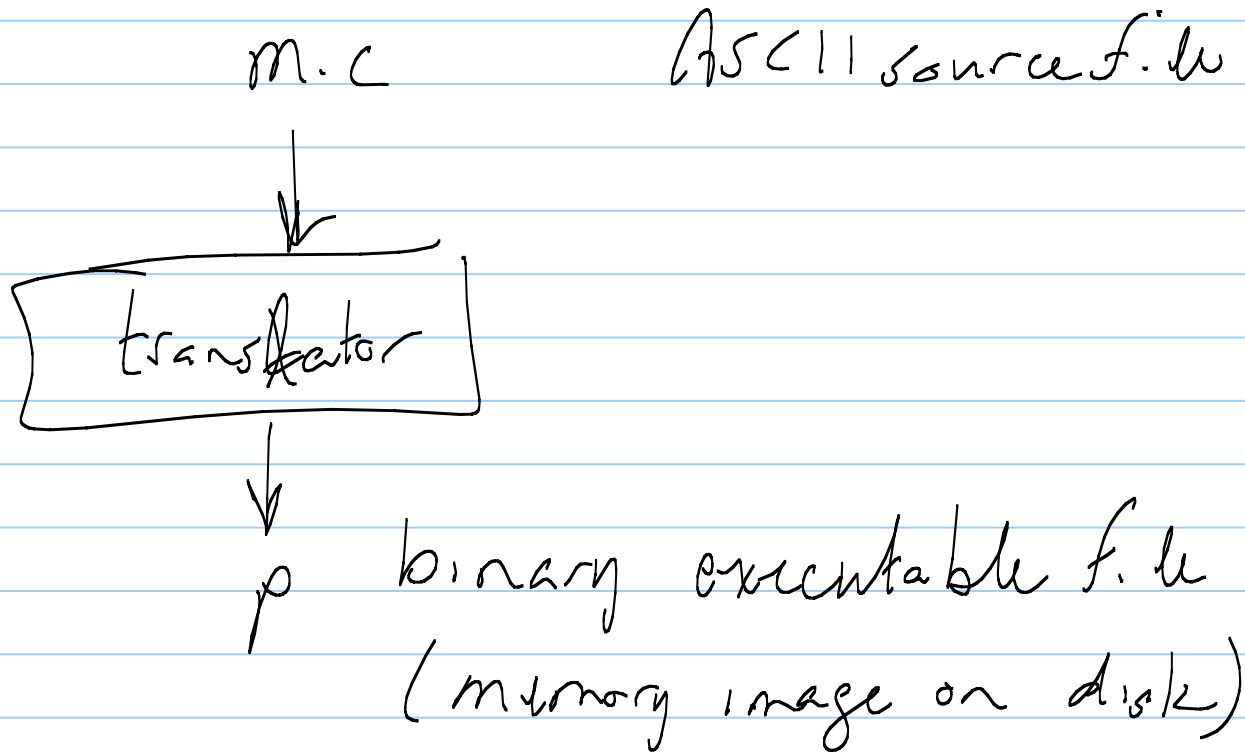


# Linking

Note Title

10/29/2007

Simple translation will work



problems

- efficiency

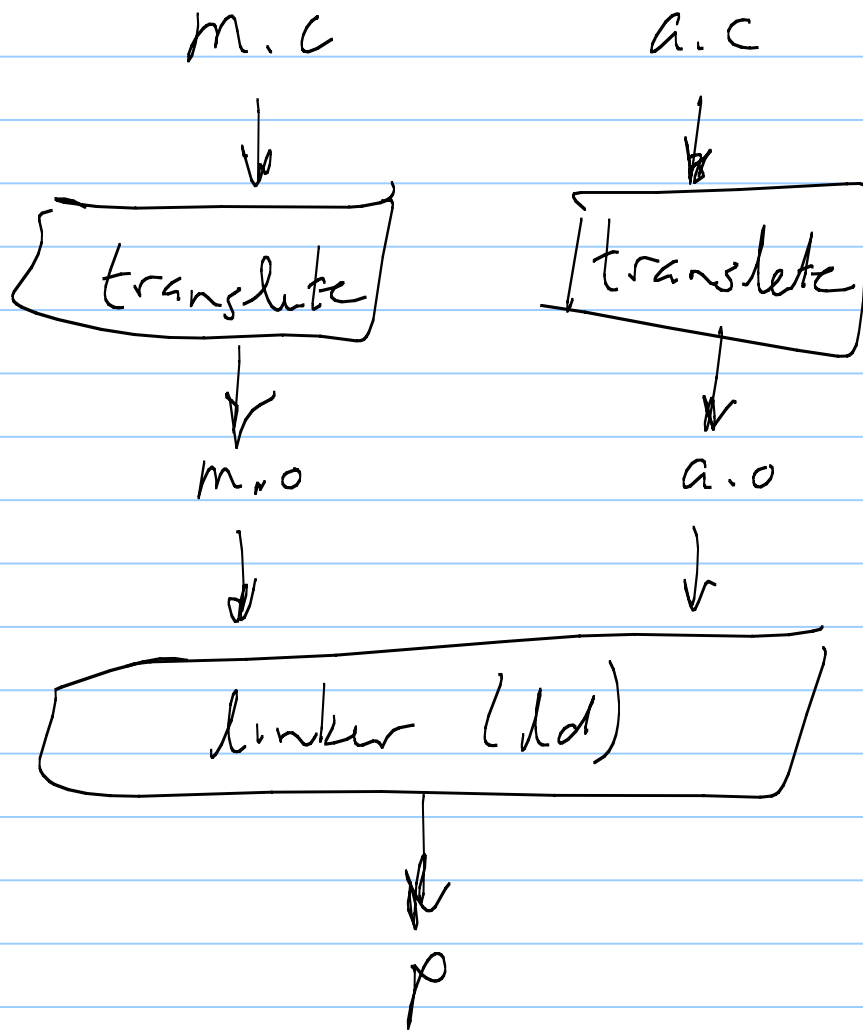
small change to source req.  
complete recompilation

- modularity

hard to share common  
function (e.g. printf)

solution

linker (static linker)



object file  
relocatable

executable object file

resolves external references

symbols declared in 1 file, used elsewhere  
relocate symbols from relative loc. in .o  
file to new absolute positions

update all references to relocated symbols

can be code:      a() //symbol a

data:    int \*xp = &x;    //symb x

## Executable and Linkable Format (ELF)

std. format for many Unix-style OS

.o      relocatable obj files

executable binaries

.so      shared object files

## Contents of ELF file!

ELF header	magic number, file type, machine
prog. header	virtual mem. addr., segments
• text section	executable code
• data section	initialized data
• bss section	uninit data
• symtab section	global symbols
• rel. txt "	relocation info for code
• rel. data "	" " " data
• debug "	debugging symbol table (-g)
• line "	maps line #s to machine code
• strtab "	constant strings table
section header table	location of sections within ELF

m.c      def of e  
          ↓  
          local symbol

```
int e = 7;  
  
int main() {  
    int r = a();  
    exit(0);  
}
```

ref to external symbol

ref to external symbol a

a.c

```
extern int e;  
  
int xep = e;  
int x = 15;  
int y;  
  
int a() {  
    return (xep + x + y);  
}
```

ref to external e

def of local symbol

refs to local symbols x, y

program symbols are "strong" or "weak"

- strong: procedure names and initialized global variables

- weak: uninitialized global variable

p1.c

→ int foo = 5;

Strong → p1() {  
}

p2.c

int foo; ← weak

p2() {  
} ← strong

## Linking rules

1. A strong symbol can appear only once
2. A weak symbol can be overridden by a strong symbol of the same name
  - refs to weak symbol resolve to strong sym.
3. If there are multiple weak symbols, the linker can pick arbitrary one.

```
int x  
p1() {}
```

```
p1() {}
```

error!

```
int x;  
p1() {}
```

```
int x;  
p2() {}
```

```
int x;  
int y;  
p1() {}
```

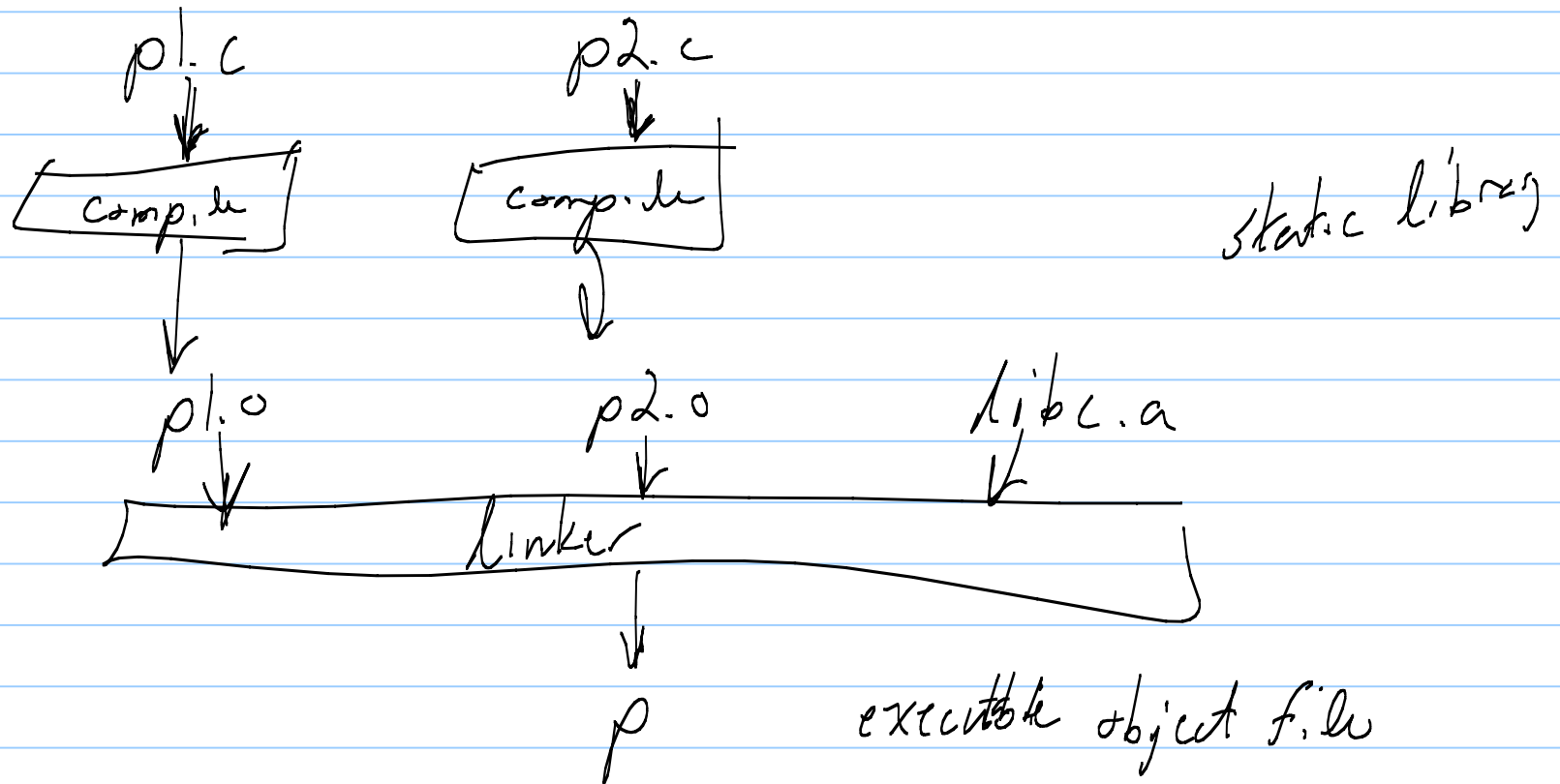
```
double x;  
p2() {}
```

Nasty

```
int x=7  
int y=7  
p1() {}
```

```
double x;  
p2() {}
```

Evil!



includes only functions in libc that are ref. by p1, p2

